Knowledge Graph Aided Network Representation and Routing Algorithm for LEO Satellite Networks

Chenxi Li, Wenji He, *Student Member, IEEE*, Haipeng Yao, *Senior Member, IEEE*, Tianle Mai, *Student Member, IEEE*, Jingjing Wang, *Senior Member, IEEE*, and Song Guo, *Fellow, IEEE*

***Abstract*—The compelling applications of Low earth orbit (LEO) satellite networks in our daily lives have been witnessed in recent years, ranging from weather forecasts to military monitoring. LEO satellite networks have grown in importance as a complement to terrestrial networks aiming at delivering global, ubiquitous communication. However, due to the LEO network topology keeps changing dynamically, the development of efficient routing algorithms becomes one of the challenges in the LEO network. Traditional routing policies hardly solve the rerouting problem caused by link switchover and the high calculation cost caused by large-scale irregular satellite topology. In this paper, we propose a knowledge graph-aided representation of satellite network topologies and routing architecture to optimize path selection and calculation cost for lower packet loss ratio and average delay. Moreover, we generate a routing policy by predicting potential relations between data packets and nodes to select the best relay nodes with the maximum probability of forwarding relations. Finally, extensive simulations are performed to evaluate the performance and availability of our proposed algorithm.**

***Index Terms*—LEO satellite networks, routing policy, network representation learning (NRL), knowledge graph (KG).**

I. Introduction

WITH the significant improvement of the broadband satellite network, it provides both global coverage and a wide variety of data communication services of different types and service quality. The constellation network composed of low earth orbit (LEO) satellite clusters has attracted more and more attention due to its low orbit, small space transmission loss, miniaturization of the ground terminal, and low power [1]. With the growth of data transportation services among global users, the higher quality of service (QoS) demands of satellite networks bring the challenge to the design of the routing algorithms [2], [3].

The routing algorithm, as one of the core technology for LEO satellite networks to carry Internet datagram communication, is important for satellite network representation and efficient real-time transmission of information. Generally, a routing algorithm consists of two stages, i.e. the network modeling stage and the routing calculation stage [4]. The network modeling stage abstracts the network topology and inner structure into a unified data structure, such as weighted adjacency matrices, and the routing calculation stage takes advantage of the abstracted model to find out the near-optimal routes for meeting QoS requirements [5]. In recent years, an increasing number of routing algorithms have been proposed to deal with the dynamics and QoS requirements of LEO satellite networks. The virtual topology strategy is proposed for complex network structures and changeable network topologies, which are stored in each satellite in fragments [6]. Abstraction and local storage of the routing information avoid numerous interactions and computations, while the satellites need enough storage capacity. Additionally, virtual node methods take advantage of the periodicity of dynamic topologies, which is suitable for the simple LEO topology with good coverage [7]. The virtual node strategy saves storage space, while the utilization rate of the feeder link is low. Existing network representation methods are difficult to take scalability, efficiency, and accuracy into account at the same time. However, the aforementioned routing strategies hardly solve rerouting problems caused by link switchover and bring heavy computational overhead in large-scale irregular satellite topologies [8].

Recently, Network Representation Learning (NRL) aims to vectorize the representation of network vertices by embedding them into a low-dimensional space. Useful to assist the analysis and optimization of networks, network representation is used for basic network modeling [9]. Then, vector-based machine learning (ML) methods can be simply used for network routing calculation tasks. Compared to discrete adjacency matrices, NRL can both capture the node's neighboring connectivity and represent other complex and high-order structure relations, such

Manuscript received 4 November 2021; revised 24 August 2022; accepted 25 November 2022. Date of publication 30 November 2022; date of current version 18 April 2023. This work was supported in part by Artificial Intelligence and Smart City Joint Laboratory (BUPT-TGSTII) under Grant B2020001, in part by Future Intelligent Networking and Intelligent Transportation Joint Laboratory (BUPTCTTIC) under Grant B2019007, and in part by Intelligent Network Joint Laboratory (BUPTIN) under Grant B2021006. The work of Jingjing Wang was supported in part by the National Natural Science Foundation of China under Grant 62222101, in part by the Young Elite Scientist Sponsorship Program by the China Association for Science and Technology under Grant 2020QNRC001, and in part by the Fundamental Research Funds for the Central Universities. The review of this article was coordinated by Dr. Tomaso De Cola. *(Corresponding author: Haipeng Yao.)*

Chenxi Li, Wenji He, Haipeng Yao, and Tianle Mai are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lichenxi.bupt@foxmail.com; hewenji@bupt.edu.cn; yaohaipeng@bupt.edu.cn; machealmai@gmail.com).

Jingjing Wang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: drwangjj@buaa.edu.cn).

Song Guo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China, and also with The Hong Kong Polytechnic University Shenzhen Research Institute, Shenzhen 518057, China (e-mail: song.guo@polyu.edu.hk).

Digital Object Identifier 10.1109/TVT.2022.3225666

as paths and high-order proximity [10]. In NRL, the knowledge graph (KG) technique is a kind of famous semantic network analysis algorithm [11], which is capable of predicting potential relations between different pieces of knowledge. It has been successfully applied in a range of compelling applications, such as semantic parsing, information extraction, and question answering [12], [13], [14]. Regarding routers, links, and data packets as different pieces of knowledge in KG, we can explore hidden similarities between new packets and former ones, and determine routing strategies relying on KG relation prediction schemes. When designing a routing algorithm, we introduce the Software Defined Networking (SDN) architecture to achieve the flexible control and efficient routing strategies of LEO satellite systems [15], [16], [17]. We consider the adaptive topology of time-varying LEO networks, load balancing, and fault tolerance. Additionally, the routing policy should not be too complex because of the limited cache and processing capabilities of satellite nodes. It can be observed that the network controller needs to steer traffic data of nodes and links as well as to meet the user's QoS constraints, such as end-to-end transmission delay, link throughput, etc. As far as we know, this is the first work using KG technology in LEO satellite network optimization. Our key contributions include:

- We propose a novel KG-aided architecture for LEO network representation to transform non-Euclidean structured network topologies and attributes into Euclidean structured low-dimensional vectors. Specifically, network entities such as nodes, links, and data packets with their various relations are embedded into vectors with the aid of mini-batch gradient descend (MBGD) [18], which improves network scalability and reduces representation complexity.
- Based on the aforementioned low-dimensional vectors, we design a KG-aided routing optimization algorithm considering both interplanetary link resources and performance requirements. For each packet, the routing policy is calculated by a consecutive selection of nodes and links from the source node to the destination node. With relation predictions, the relay nodes and links can be selected relying on a loss function considering their relational vectors and attributes.
- Extensive experiments are conducted for evaluating the performance of our proposed algorithm. Simulation results show that our algorithm outperforms several benchmarks in terms of both packet loss ratio and average delay.

The rest of this paper is organized as followed. In Section II, we review the related work about the routing algorithms in LEO satellite networks as well as NRL schemes. In Section III, we propose a KG-based routing model that aided the optimization metrics and representation constraints. In Section IV, the KG-aided network representation method is put forward. In Section V, we propose a routing algorithm optimization based on the vectorized representation. In Section VI, simulation results and performance evaluation are provided, followed by our conclusions and future works in Section VII.

II. Related Work

In this section, we overview routing algorithms in LEO satellite networks and NRL algorithms.

A. Routing Algorithms in LEO Satellite Network

LEO satellite network has the characteristics of continuous high dynamic topology and high inter-satellite packet loss rate. In [19], M. Werner et al. proposed the discrete-time dynamic virtual topology routing algorithm to divide the satellite time-varying topological network equally into several fixed topologies, which avoided the negative impact of satellite motion on routing calculation, but it was difficult to effectively control the link failure and network congestion. In [20], Hashimoto et al. designed a routing algorithm for low-orbit satellite networks based on IP. In [21], Ekici et al. proposed a DRA algorithm based on virtual nodes to realize topology control, but only considered the local information of satellite nodes. To solve the reroute problem caused by link switchover, performances such as resource allocation and QoS have been considered [22], [23]. According to the division of the coverage domain and the separation of the satellite segment, the hop-constrained adaptive routing mechanism with load balancing capability was proposed [24]. To better solve the issue of frequent link switching between nodes caused by high-speed topology changes in LEO satellite networks, a dynamic routing algorithm is usually adopted, while the routing protocol costs more [25], [26]. However, routing and forwarding policies are influenced by each satellite, which results in network coupling and difficulties in network management. Researchers turned their focus on software-defined networking (SDN) which enables flexible network management and control, especially for distributed low-layer network entities [27]. In [28], an SDN architecture consisted of a master controller and several slave controllers to realize distributed control with centralized management. The satellites are responsible for simply forwarding data obeying the flow table and informing the controllers of the network status [29], [30].

Traditionally, routing algorithms can be roughly divided into three categories according to their algorithm patterns, i.e. static, heuristic and AI approaches. The complexity and scalability of these three routing algorithms are compared in Table I, and some representative algorithms are listed. To elaborate, static routing algorithms determine routing strategies by explicit calculations, e.g. shortest path algorithms, which have been commonly adopted in small-scale networks. In [31], Dijkstra proposed the shortest path algorithm that calculated the shortest paths from a single node to the others by selecting the shortest link each time. In [32], Floyd proposed the shortest path algorithm that can calculate the shortest paths for all pairs of nodes by regarding each node as a relay. However, they cannot respond to the change of network state and perform congestion avoidance. Besides, heuristic approaches aim at calculating optimal or sub-optimal routing schemes through iterative calculations, such as ant colony optimization (ACO), genetic algorithm (GA), and particle swarm optimization (PSO). By contrast, AI-based algorithms rely on ML techniques for the sake of determining routing schemes, such as reinforcement learning [29] and deep learning (DL) [40]. In Table I, we compare these algorithms in terms of their complexity and extensibility and then list some representative algorithms of different types. Despite successful applications arising, AI-based routing algorithms have not achieved the expected objectives of scalability and robustness.

TABLE I
COMPARISON OF STATIC, HEURISTIC AND AI APPROACHES ON ROUTING

Algorithm type	Method type	Time Complexity	Space Complexity	Extensibility	Algorithms
Static	Explicit calculation	Low	Low	Low	Dijkstra algorithm [31], Floyd algorithm [32], Bellman-Ford (BF) [33], DT-DVTR [19]
Heuristic	Iterative optimization	High	Medium	Low and high cost	ACO-R [34], GA-SDN [35], PSOPR [36]
AI	Machine Learning	High (training), low (after training)	High	Scalable	MLQU [37], SeDaTiVe [38], DROM [39]

B. Network Representation Learning

As mentioned before, NRL can be used to embed network vertices into low-dimensional vectors and preserve network topologies as well as other attributes of each vertex. In [41], Yao et al. proposed a QoS routing algorithm by first encoding network routers as vectors and then calculating route paths by table queries. Zhang et al. [9] provided a comprehensive review of the state-of-the-art NRL techniques and useful applications such as KG, which is a powerful technique for relation prediction in databases. Wang et al. [11] surveyed the whole framework, explicit model design, and classical training procedures of KG architecture. Bordes et al. [42] proposed the so-called TransE by first extracting relational information in the form of triples, consisting of a head entity, a tail entity, and a relation between them. Then they embedded knowledge entities with relation information into low-dimensional vectors by minimizing a loss function on triples, where fact triples had lower values than false ones. Additionally, Wang et al. [43] proposed a modified version TransH, which introduced a hyperplane for each relation representation in the network. By calculating projections on different hyperplanes, TransH solves the embedding of reflexive, one-to-many, many-to-one, and many-to-many relations. NRL can be used to capture both revealed information, such as connectivity and topology, and latent information such as paths and proximity of the network, which can help route optimization processes and has great potential if combined with powerful AI-based routing algorithms.

III. SYSTEM MODEL

In this section, we model our KG-aided network representation of LEO satellite networks and routing algorithms. As mentioned before, SDN architecture can minimize the satellite's onboard processing cost and have superiority in network integration [44]. Thus, as shown in Fig. 1, a knowledge graph structure can be applied to the control plane consisting of GEO satellites and several LEO satellites. The control plane detects the dynamic topology based on NRL aided by a knowledge graph and computes routing policies. LEO satellites in the data plane do not need to carry out route calculation, but only need to maintain the flow table, which is executed according to the controller's policy. Then, the routing algorithms are executed and calculated on the control plane, which is able to abstract

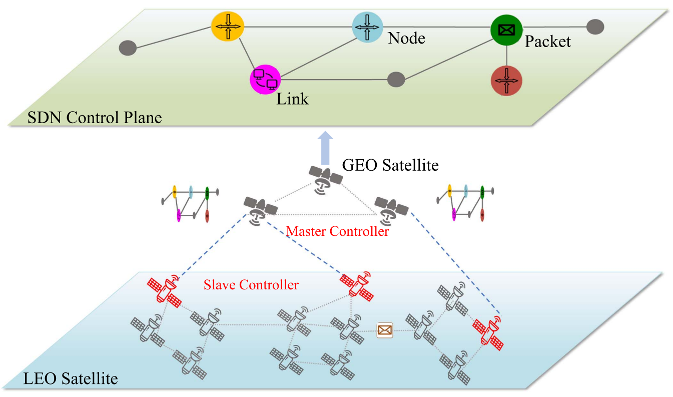

Fig. 1. A knowledge graph aided LEO satellite network model.

and model the network nodes, links, and data packets. In order to manage the network more flexibly and update routing paths more efficiently, we adopt a master controller of the GEO satellite and multiple slave controllers of LEO satellites in our model. Consider a network with a set of LEO satellite network elements and relations $G = \{V, E, P, R\}$, where V is routers (nodes), E is links between nodes, P is data packets, and R is different kinds of relations between them. For each network element $x_i \in \{V \cup E \cup P\}$, we define a k-dimensional normalized vector to represent its similarities to the other elements on the control plane. The satellite cycle is divided into n time slices ΔT by a time-sharing method. In each time slice, these vectorized elements are referred to as entities in our KG architecture, which is depicted as colored circles in Fig. 1. Relations $r_i \in R$ represent the relationship between network elements, e.g. connections between nodes and links, and data packets forwarding by nodes, as the edges in the control plane in Fig. 1. We define a k-dimensional normalized vector to represent a relation, and another k-dimensional vector for the hyperplane when different kinds of entities are involved. In our model, we abstract six kinds of relations to better illustrate the network, which will be further explained in Section IV. In addition to relational information such as entities and relations, we also define attribute information of an element to represent its storage and processing capability, which can be precisely designed for different network targets and functions. In this paper, since we mainly focus on a routing optimization problem, specifically, we take scalability, efficiency, and accuracy as optimization targets.

- Scalability: The dimensionality of the matrix becomes larger as satellite networks expand, which may slow down the learning and training processes for routing calculation.
- Efficiency: In a large-scale network, most of the nodes are connected to only a few neighbor nodes. Thus, the adjacency matrix is sparse enough with most of the elements being 0. It is inefficient to perform optimization algorithms on sparse matrices, which always requires additional dimensionality reduction pre-processes.
- Accuracy: The rows and columns of the matrices are sorted by the indexes of the nodes, containing little information about the network state. Changing the order of rows or columns in these matrices can result in different matrices without changes in network topologies. It means that matrix representation is inaccurate for representing a network.

The following elements and attributes are taken into consideration.

- Node: maximum queue length, current queue length, and transmission rate.
- Link: maximum bandwidth, current bandwidth, and delay.
- Data packet: packet size and delay requirement.

To evaluate our algorithm, we use the following three evaluation metrics to evaluate the performance of our routing algorithm [37].

- Packet Loss Ratio: packet loss indicates that a data packet fails to arrive at the destination node within the required delay. In our work, we assume that the channel noise and signal fading are not strong enough to affect the transmission of data packets. Thus, packet loss may occur in the buffer of a node when a packet is transferred to a node with maximum queue length, or when it fails to reach its destination node within its required delay. It can then be defined as

$$\text{Packet Loss} = \frac{\text{Number of lost data packets}}{\text{Number of send data packets}}. \quad (1)$$

- Throughput: network throughput measures the capability of a network to transmit data packets per unit of time, which is generally determined by network topologies, available resources, and routing strategies. In this paper, we define the average throughput of LEO networks as

$$\text{Throughput} = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} \text{Data packets transmitted}}{\text{Unlost packets T}}. \quad (2)$$

- Average delay: average delay refers to the average transmission time from the source node to the destination node of all data packets which have been successfully routed. A higher average delay has a negative influence on real-time network applications, such as online video conferences and electronic sports. Besides, it also reduces the network resource utilization, since a data packet tends to take up buffer or bandwidth resources for a longer period, which may further lead to a higher packet loss ratio. In this paper, we define the average delay as

$$\text{Average Delay} = \frac{\sum_{p\in P} delay_p}{\|P\|}, \quad (3)$$

where $\|\cdot\|$ is the cardinality of a set.

The KG model has the following constraints

$$0 \le QL_v(t) \le QL_v^m, \forall t, \forall v \in V, \quad (4)$$

$$0 \le BW_l(t) \le BW_l^m, \forall t, \forall l \in E, \quad (5)$$

$$\frac{\left|\boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{d_r}\right|}{\|\boldsymbol{d_r}\|_2} = 0, \forall r \in R, \quad (6)$$

$$\sum_{v_i \in Path(p)} D_{v_i} + \sum_{l_i \in Path(p)} D_{l_i} \le D_p^{req}, \forall p \in P, \quad (7)$$

$$\begin{aligned} \|\boldsymbol{w_r}\|_2 &= 1, \forall r \in R, \\ \|\boldsymbol{v_r}\|_2 &= 1, \forall v \in V, \\ \|\boldsymbol{l_r}\|_2 &= 1, \forall l \in E, \\ \|\boldsymbol{p_r}\|_2 &= 1, \forall p \in P. \end{aligned} \quad (8)$$

where $\|\cdot\|_2$ is the $L2$ norm of a vector. Constraints (4) and (5) ensure that the queue length of each node and the bandwidth of each link are both less than their maximum. (6) means that $\boldsymbol{w_r}$ and $\boldsymbol{d_r}$ are orthogonal, so that $\boldsymbol{d_r} \in$ hyperplane W_r. Besides, inequality (7) is the delay requirement constraint that the route path should satisfy. (8) normalize the entity vectors and relation vectors to reduce calculation complexity. We list the notations of this paper with explicit explanations in Table II.

TABLE II
NOTATIONS AND SYMBOLS

Notations & symbols	Definitions
V, E, P, R	Set of nodes, links, data packets and relations
$\boldsymbol{v_r}$, $\boldsymbol{v_c}$	Relation and attribute vector of node $v \in V$
$\boldsymbol{l_r}$, $\boldsymbol{l_c}$	Relation and attribute vector of link $l \in E$
$\boldsymbol{p_r}$, $\boldsymbol{p_c}$	Relation and attribute vector of data packet $p \in P$
W_r, $\boldsymbol{w_r}$	Hyperplane and its unit normal vector of $r \in R$
$\boldsymbol{d_r}$	Vector of $r \in R$
QL_v^m, $QL_v(t)$	Maximum queue length and current queue length at time t of node $v \in V$
BW_l^m, $BW_l(t)$	Maximum bandwidth and current bandwidth at time t of link $l \in E$
D_v, D_l	Delay of node $v \in V$ and link $l \in E$
D_p^{req}	Delay requirement of packet $p \in P$
Δ, Δ'	Set of positive and negative triples
$\Delta'_{\{h,r,t\}}$	Set of negative triples generated from positive triple $\{h, r, t\}$
$\mathbb{R}^k$	k-dimensional vector space

IV. Knowledge Graph Aided Network Representation

The main idea of our KG-aided routing algorithm is to 1) transform the non-Euclidean-structured network into an AI-friendly Euclidean-structured model by embedding network entities into low-dimensional vectors, and 2) predict relations between new data packets and existing network entities to decide

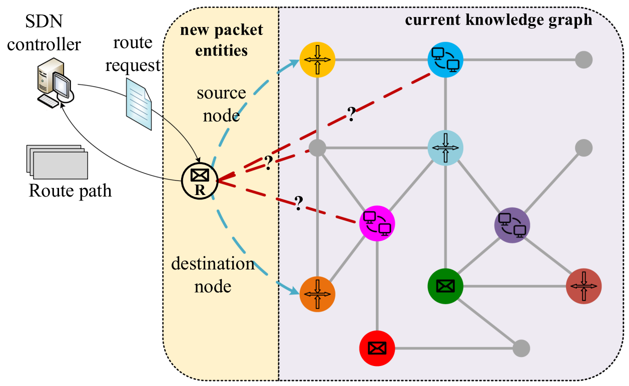

Fig. 2. Route calculation and decision in the knowledge graph.

optimal routes, as shown in Fig. 2. In this section, we first introduce our KG-aided network representation algorithm.

A. General Framework

The first stage of our algorithm is the KG-aided network representation, i.e. vectorizing the network entities and relations. The global SDN controller first extracts relational information of network entities and relations from the topologies of triples consisting of two entities and one relation. Then we use these triples to train a KG embedding model that transforms entities and relations into low-dimensional vectors, i.e. $\boldsymbol{v_r}$, $\boldsymbol{l_r}$, $\boldsymbol{p_r}$, $\boldsymbol{w_r}$ and $\boldsymbol{d_r}$ by minimizing the loss function of the triples. The second stage of our algorithm is the calculation of routing schemes. The controller determines routing strategies by calculating similarities and predicting possible relations between the users' requests and the network elements [45]. To be specific, we first calculate an entity vector for the new data packet according to its known relational information, such as source node, destination node, and similarities to other data packets. Then we calculate a feasible route by first predicting its possible relations between other node entities and link entities, and then selecting the optimal relay nodes.

B. Knowledge Graph Embedding

KG embedding is one of the critical processes in KG-aided network representation, which embeds network entities and relations into low-dimensional vectors. It consists of two consecutive stages, entity and relation extraction stage and vector embedding stage.

1) Entity and Relation Extraction: In this stage, the controller collects network state and topology for KG embedding. Different from traditional NRL methods which only focus on vertices, KG architecture is more effective to represent relational information since it considers both entities and relations. The data structure extracted from the network is in the form of ordered triples (h, r, t), where h, t, and r represent the head, tail, and their relations, respectively. A positive triple $(h, r, t) \in \Delta$ indicates that there is a relation r between entity h and t, while a negative one $(h, r, t) \in \Delta'$ indicates the opposite. In this paper, the entities can be classified into three categories, node, link, and the data packet, respectively. The predefined relations are shown in Table III.

TABLE III
TYPE OF RELATIONS

Relations	Definitions	Head	Tail
$Connect$	Direct connection	Node	Link
$NodeConnect$	Adjacent nodes	Node	Node
$SourceOf$	Source node	Node	Packet
$DestinationOf$	Destination node	Node	Packet
$NodePassedBy$	Node on the route	Node	Packet
$LinkPassedBy$	Link on the route	Link	Packet

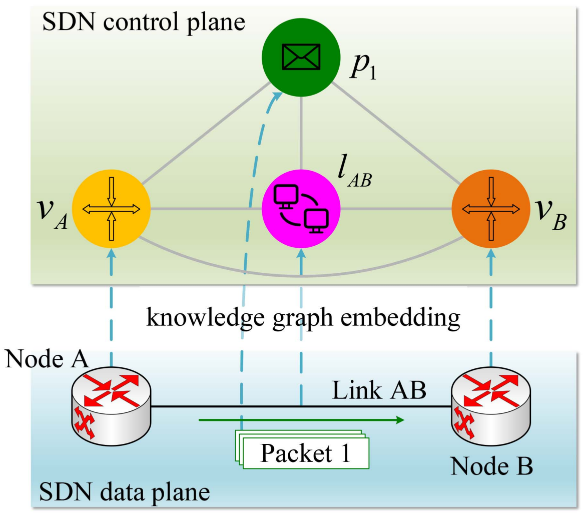

Fig. 3. According to the extraction rules we defined, the following triples can be extracted: (v_A, Connect, l_{AB}), (v_B, Connect, l_{AB}), (v_A, NodeConnect, v_B), (v_B, NodeConnect, v_A), (v_A, SourceOf, p_1), (v_B, DestinationOf, p_1), (l_{AB}, LinkPassedBy, p_1).

We design two types of relations to better represent topological information about the substrate network. $Connect$ describes the relations between nodes and adjacent links, which is similar to the first-order proximity in the social network [46]. $NodeConnect$ describes the relations between adjacent nodes, which is similar to the second-order proximity [46]. Hence the entity vectors can capture relational information in a larger neighbor. Fig. 3 is a demo of two adjacent nodes A, and B with a link AB, and a data packet 1 transmitting from node A to node B.

2) Vector Embedding: In this paper, we propose a modified TransH algorithm to calculate entity and relation vectors in our KG-aided network modeling [43]. TransH is an improved algorithm of TransE [42] by defining a hyperplane for each relationship so that one entity can own various vectorized expressions in different relations. Thus, the model is more capable of handling one-to-many, many-to-one, many-to-many relations, and reflexive relations, which is common in SDN representations. For example, a node may connect to many other adjacent links, while a data packet can transmit on different nodes and links.

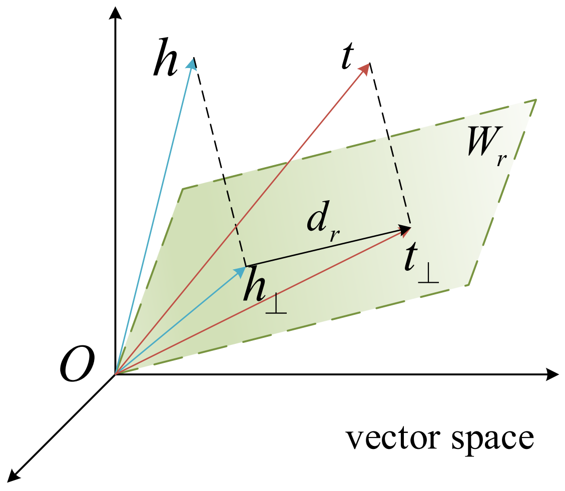

Fig. 4. An example of TransH and its hyperplane.

To be specific, we model each entity as a k-dimensional vector and each relation $r \in R$ as a combination of a hyperplane W_r with its normal vector $\boldsymbol{w_r}$, and a relation vector $\boldsymbol{d_r}$ on the hyperplane. As shown in Fig. 4, if h and t satisfy relation r, i.e. $(h, r, t) \in \Delta$, the projection $\boldsymbol{h}_\perp$ of vector $\boldsymbol{h}$ on W_r and the projection $\boldsymbol{t}_\perp$ of vector $\boldsymbol{t}$ on W_r satisfy

$$\boldsymbol{h}_\perp + \boldsymbol{d_r} = \boldsymbol{t}_\perp, . \tag{9}$$

Note that here we only plot a three-dimensional vector space for visualization. The real dimension of the vector space is usually up to 50 or more, based on the actual topology complexity.

From the mathematical definition of vector projection, we have

$$\boldsymbol{h}_\perp = \boldsymbol{h} - \boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{h} \boldsymbol{w_r}, \tag{10}$$

$$\boldsymbol{t}_\perp = \boldsymbol{t} - \boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{t} \boldsymbol{w_r}. \tag{11}$$

Thus, according to (9–11), for a positive triple $(h, r, t) \in \Delta$, we have

$$\| \left(\boldsymbol{h} - \boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{h} \boldsymbol{w_r}\right) + \boldsymbol{d_r} - \left(\boldsymbol{t} - \boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{t} \boldsymbol{w_r}\right) \|_2 \approx 0. \tag{12}$$

On the contrary, for a negative triple $(h,' r,' t') \in \Delta'$, we have

$$\| \left(\boldsymbol{h'} - \boldsymbol{w_{r'}}^{\mathrm{T}} \boldsymbol{h'} \boldsymbol{w_{r'}}\right) + \boldsymbol{d_{r'}} - \left(\boldsymbol{t'} - \boldsymbol{w_{r'}}^{\mathrm{T}} \boldsymbol{t'} \boldsymbol{w_{r'}}\right) \|_2 \gg 0. \tag{13}$$

Hence, the loss function of the triple (h, r, t) in our model can be represented as

$$f(h, r, t) = \| \left(\boldsymbol{h} - \boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{h} \boldsymbol{w_r}\right) + \boldsymbol{d_r} - \left(\boldsymbol{t} - \boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{t} \boldsymbol{w_r}\right) \|_2^2. \tag{14}$$

It then can be proved that a lower $f(h, r, t)$ indicates that (h, r, t) is more likely to be a positive triple reflecting a true relation, while a higher $f(h, r, t)$ tends to represent a false one.

3) Training of Knowledge Graph Embedding Model: To classify positive triples and negative ones, we train a KG embedding model. However, negative triples usually outnumber positive ones in actual networks, especially large ones, where nodes are only connected to a few adjacent nodes and links, and isolated from most of the others. If all the triples are directly input to the KG embedding model, it is hard for the model to learn the features of different types from these skewed data. Thus, in this paper, we generate a dataset by randomly generating a small number of negative triples for each positive triple to avoid unbalanced training data. We denote the proportion of negative triples to positive ones as η. Furthermore, when generating negative triples, either the head or tail entity is replaced, while the relation keeps the same so that the model will be more capable of classifying similar triples. That is, for a positive triple (h, r, t), the generated negative one is either $(h,' r, t) \notin \Delta$ or $(h, r, t') \notin \Delta$. In the following paragraphs, we simply denote negative triples as $(h,' r, t')$ for convenience, but it does not mean that both head and tail entities are replaced.

In our model, we make use of the linear hinge loss as the loss function, which can be given by

$$Loss = \sum_{\substack{(h,r,t)\in\Delta, \\ (h,'r,t')\in\Delta'_{(h,r,t)}}} \left[f(h, r, t) - f(h,' r, t') + \gamma\right]_+, \tag{15}$$

where γ is the margin of positive and negative triples. There exists $[x]_+ = \max(0, x)$. Thus, the objective and constraints of our training process is

$$\min_{\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}} \sum_{\substack{(h,r,t)\in\Delta, \\ (h,'r,t')\in\Delta'_{(h,r,t)}}} \left[f(h, r, t) - f(h,' r, t') + \gamma\right]_+$$
$$\text{s.t.} \quad (6), (8). \tag{16}$$

Since constraint (6) is usually difficult to satisfy accurately, we introduce a weight hyper-parameter C_1 to combine it with the objective function. Then, the original problem is transformed to

$$\min_{\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}} \sum_{\substack{(h,r,t)\in\Delta, \\ (h,'r,t')\in\Delta'_{(h,r,t)}}} \left[f(h, r, t) - f(h,' r, t') + \gamma\right]_+ + C_1 \sum_{r\in R} \left(\boldsymbol{w_r}^{\mathrm{T}} \boldsymbol{d_r}\right)^2$$
$$\text{s.t.} \quad (8). \tag{17}$$

We adopt the modified batch gradient descent (MBGD) algorithm to calculate the minimum of formula (17) and normalize the vectors. MBGD is a modified combination of the batch gradient descent (BGD) algorithm and the stochastic gradient descent (SGD) algorithm. BGD is a traditional gradient descent algorithm that updates parameters by calculating the gradient on all training data, which is time-consuming for large datasets. On the contrary, SGD updates parameters according to the gradient on a single training data each time, which calculates faster but converges more slowly since a single training triple may cause (17) to rise. MBGD, on the other hand, updates parameters according to the gradient on a mini-batch of training data, which is a random sample of the dataset. In a single epoch, the parameters are updated several times. Usually, MBGD calculates the gradient faster than BGD and converges faster than SGD.

Denote L as the objective function in formula (17). Then, we can calculate the gradient of L as

$$\nabla L\left(\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{w_r}, \boldsymbol{d_r}\right) = \left\{\frac{\partial L}{\partial \boldsymbol{h}}, \frac{\partial L}{\partial \boldsymbol{t}}, \frac{\partial L}{\partial \boldsymbol{w_r}}, \frac{\partial L}{\partial \boldsymbol{d_r}}\right\}. \tag{18}$$

Algorithm 1: Training of KG Embedding Model.

```
Input: Training set $\Delta_{train}$, $G = \{V, E, P, R\}$, margin $\gamma$, embedding dimension $k$, proportion parameter $\eta$, learning rate $\alpha$, training times $epochs$, mini-batch size $b$, weight parameter $C_1$.
Output: Vectors of entities and relations $\boldsymbol{v_r}$, $\boldsymbol{l_r}$, $\boldsymbol{p_r}$, $\boldsymbol{w_r}$, $\boldsymbol{d_r}$.
Initialization
1:  for each $\boldsymbol{v_r} \in V, \boldsymbol{l_r} \in E, \boldsymbol{p_r} \in P$
2:      $\boldsymbol{v_r}, \boldsymbol{l_r}, \boldsymbol{p_r} = uniform\left(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right)$
3:      $\boldsymbol{v_r} = \frac{\boldsymbol{v_r}}{\|\boldsymbol{v_r}\|}, \boldsymbol{l_r} = \frac{\boldsymbol{l_r}}{\|\boldsymbol{l_r}\|}, \boldsymbol{p_r} = \frac{\boldsymbol{p_r}}{\|\boldsymbol{p_r}\|}$
4:  end for
5:  for each $r \in R$ do
6:      $\boldsymbol{w_r}, \boldsymbol{d_r} = uniform\left(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right)$
7:      $\boldsymbol{w_r} = \frac{\boldsymbol{w_r}}{\|\boldsymbol{w_r}\|}, \boldsymbol{d_r} = \frac{\boldsymbol{d_r}}{\|\boldsymbol{d_r}\|}$
8:  end for
Training
9:  while $epoch < epochs$: do
10:     for each $batch \in batchNum$ do
11:     $S_{batch} = sample\left(\Delta_{train}, b\right)$ // sample a mini-batch of $b$ triples from the training set
12:         $T_{batch} = \varnothing$
13:         for each $(h, r, t) \in S_{batch}$ do
14:             for $i = 1, 2, \ldots, \eta$ do
15:                 generate $(h,' r, t') = sample\left(\Delta'_{(h,r,t)}\right)$
16:                 $T_{batch} = T_{batch} \cup \{((h, r, t), (h,' r, t'))\}$
17:             end for
18:         end for
19:         update $\Theta = \Theta - \alpha \sum_{((h,r,t),(h,'r,t')) \in T_{batch}} \nabla\left\{[f(h,r,t) - f(h,'r,t') + \gamma]_+ + C_1 \sum_{r \in R}(\boldsymbol{w_r}^{\mathrm{T}}\boldsymbol{d_r})^2\right\}$ // where $\Theta = \{\boldsymbol{v_r}, \boldsymbol{l_r}, \boldsymbol{p_r}, \boldsymbol{w_r}, \boldsymbol{d_r}\}$
20:         for each $\boldsymbol{v_r}$, $\boldsymbol{l_r}$, $\boldsymbol{p_r}$, $\boldsymbol{w_r}$ do
21:             $\boldsymbol{v_r} = \frac{\boldsymbol{v_r}}{\|\boldsymbol{v_r}\|}, \boldsymbol{l_r} = \frac{\boldsymbol{l_r}}{\|\boldsymbol{l_r}\|}, \boldsymbol{p_r} = \frac{\boldsymbol{p_r}}{\|\boldsymbol{p_r}\|}, \boldsymbol{w_r} = \frac{\boldsymbol{w_r}}{\|\boldsymbol{w_r}\|}$
22:         end for
23:     end for
24:     $epoch = epoch + 1$
25: end while
26: return $\boldsymbol{v_r}$, $\boldsymbol{l_r}$, $\boldsymbol{p_r}$, $\boldsymbol{w_r}$, $\boldsymbol{d_r}$
```

Thus, the training process of the KG embedding model is shown in Algorithm 1.

V. Routing Optimization

After the KG embedding process, we obtain the vectorized expressions of network entities and relations. Then we can calculate routes based on the network model. The route calculation consists of two consecutive stages in our paper, vector calculation, and route decision.

1) Vector Calculation

To utilize the relation analysis capability of the KG architecture, we first model the incoming data packet as a k-dimensional relation vector. Suppose a new data packet p_n has been received by the global controller. It contains a) relational information including source node v_s and destination node v_d, and b) attribute information including packet size and delay requirement. Thus, $(v_s, SourceOf, p_n)$ and $(v_d, DestinationOf, p_n)$ are positive triples, and relation vector $\boldsymbol{v_{s_r}}$, $\boldsymbol{v_{d_r}}$ and $\boldsymbol{p_{n_r}}$ should satisfy

$$\left(\boldsymbol{v_{s_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{v_{s_r}}\boldsymbol{w_s}\right) + \boldsymbol{d_s} - \left(\boldsymbol{p_{n_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{p_{n_r}}\boldsymbol{w_s}\right) = 0, \quad (19)$$

$$\left(\boldsymbol{v_{d_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{v_{d_r}}\boldsymbol{w_d}\right) + \boldsymbol{d_d} - \left(\boldsymbol{p_{n_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{p_{n_r}}\boldsymbol{w_d}\right) = 0, \quad (20)$$

where $\boldsymbol{w_s}$ and $\boldsymbol{d_s}$ denote the normal vector of hyperplane and relation vector of $SourceOf$ relation, while $\boldsymbol{w_d}$ and $\boldsymbol{d_d}$ denote those of $DestinationOf$ relation, respectively.

Note that (19)–(20) hardly have exact solutions in $\mathbb{R}^k (k \geq 3)$. It can be proved by combining (19) and (20), i.e.

$$\left(\boldsymbol{v_{s_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{v_{s_r}}\boldsymbol{w_s}\right) + \boldsymbol{d_s} - \left(\boldsymbol{v_{d_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{v_{d_r}}\boldsymbol{w_d}\right) - \boldsymbol{d_d}$$
$$= \left(\boldsymbol{p_{n_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{p_{n_r}}\boldsymbol{w_s}\right) - \left(\boldsymbol{p_{n_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{p_{n_r}}\boldsymbol{w_d}\right). \quad (21)$$

Since $\boldsymbol{w_s}$, $\boldsymbol{w_d}$ and $\boldsymbol{p_{n_r}}$ are k-dimensional vectors, we can rewrite (21) as non-homogeneous linear equations

$$\left(\boldsymbol{w_s}\boldsymbol{w_s}^{\mathrm{T}} - \boldsymbol{w_d}\boldsymbol{w_d}^{\mathrm{T}}\right)\boldsymbol{p_{n_r}} = \left(\boldsymbol{v_{s_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{v_{s_r}}\boldsymbol{w_s}\right)$$
$$+ \boldsymbol{d_s} - \left(\boldsymbol{v_{d_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{v_{d_r}}\boldsymbol{w_d}\right) - \boldsymbol{d_d}. \quad (22)$$

Let A denote the coefficient matrix $\boldsymbol{w_s}\boldsymbol{w_s}^{\mathrm{T}} - \boldsymbol{w_d}\boldsymbol{w_d}^{\mathrm{T}}$ and $\boldsymbol{b}$ denote the constant term on the right, i.e.,

$$A = \boldsymbol{w_s}\boldsymbol{w_s}^{\mathrm{T}} - \boldsymbol{w_d}\boldsymbol{w_d}^{\mathrm{T}}, \quad (23)$$

$$\boldsymbol{b} = \left(\boldsymbol{v_{s_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{v_{s_r}}\boldsymbol{w_s}\right) + \boldsymbol{d_s} - \left(\boldsymbol{v_{d_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{v_{d_r}}\boldsymbol{w_d}\right) - \boldsymbol{d_d}. \quad (24)$$

Thus we can rewrite (22) as

$$A\boldsymbol{p_{n_r}} = \boldsymbol{b}. \quad (25)$$

It can be proved that $rank(A) \leq 2$, since the number of vectors in its maximal linearly independent set is 2, i.e. $\{\boldsymbol{w_s}, \boldsymbol{w_d}\}$. The equality holds true if and only if $\{\boldsymbol{w_s}$ and $\boldsymbol{w_d}\}$ are linearly independent. However, the rank of the augmented matrix $rank(A, \boldsymbol{b})$ is possible to be 3 when $k \geq 3$, depending on $\boldsymbol{w_s}$, $\boldsymbol{w_d}$, $\boldsymbol{v_s}$ and $\boldsymbol{v_d}$. If $rank(A) < rank(A, \boldsymbol{b})$, (19)–(20) can't calculate specific solutions $\boldsymbol{p_{n_r}}$. Hence, in order to calculate an approximate vector representation in case of no solutions, we define the accuracy function of $\boldsymbol{p_{n_r}}$ as

$$g\left(\boldsymbol{p_{n_r}}\right) = \left\|\left(\boldsymbol{v_{s_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{v_{s_r}}\boldsymbol{w_s}\right) + \boldsymbol{d_s} - \left(\boldsymbol{p_{n_r}} - \boldsymbol{w_s}^{\mathrm{T}}\boldsymbol{p_{n_r}}\boldsymbol{w_s}\right)\right\|_2^2$$
$$+ \left\|\left(\boldsymbol{v_{d_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{v_{d_r}}\boldsymbol{w_d}\right) + \boldsymbol{d_d} - \left(\boldsymbol{p_{n_r}} - \boldsymbol{w_d}^{\mathrm{T}}\boldsymbol{p_{n_r}}\boldsymbol{w_d}\right)\right\|_2^2$$
$$+ C_2\left(\|\boldsymbol{p_{n_r}}\|_2 - 1\right), \quad (26)$$

where C_2 is the weight parameter for the normalization of $\boldsymbol{p_{n_r}}$. By minimizing $g(\boldsymbol{p_{n_r}})$ utilizing MBGD (similar to Algorithm 1), we can obtain a sub-optimal vector representation $\boldsymbol{p_{n_r}}$ for the new data packet p_n. Then we can calculate routing results according to the similarities between $\boldsymbol{p_{n_r}}$ and the existing entities in the network.

2) Route Decision

In this paper, the route decision is carried out through consecutive selecting of nodes and links, until the data packet

Algorithm 2: Route Decision Algorithm.

```
Input: $G = \{V, E, P, R\}$ and their corresponding vectors
  $\boldsymbol{v_r}, \boldsymbol{v_c}, \boldsymbol{l_r}, \boldsymbol{l_c}, \boldsymbol{p_r}, \boldsymbol{p_c}, \boldsymbol{w_r}, \boldsymbol{d_r}$, source node $v_s$, destination
  node $v_d$, delay requirement $D_p^{req}$, weight factor $\beta$.
Output: Route list $List = (v_s, \ldots, v_d)$
Initialization
1: Normalize $\boldsymbol{v_c}, \boldsymbol{l_c}, \boldsymbol{p_c}$
2: $List = (v_s)$
   Route decision
3: while $v_i \neq v_d$: do
4:   Find best next hop $v_{i+1}$
5:   $List.append(v_{i+1})$
6:   $v_i = v_{i+1}$
7: end while
8: return $List$
```

reaches its destination node. Generally, one link is connected to two different nodes, selecting the next link is equivalent to the next node when considering the optimal next hop of a packet. Hence, both link information and node information should be considered in our route decision algorithm.

Consider a routing task to decide the next hop (v_{k+1}, l_{k+1}) of a data packet p_n. As mentioned before, our algorithm aims at supporting QoS-guaranteed network transmitting services and prolonging network lifetime. Thus, we should select links with more available bandwidth resources and nodes with shorter queue length to avoid congestion at busy nodes and links. We define the selection process of the next hop as

$$\arg\min_i sim_i \times \left[\frac{\beta}{D_p'} \times \left(d_{l_i} + \frac{QL_{v_i}}{TR}\right) + (1-\beta)L_p' \times \left(\frac{1}{BW_{l_i}} + QL_{v_i}\right)\right], \quad (27)$$

where $\beta \in [0, 1]$ is a weight factor to adjust the weight of the next node and its corresponding link in the calculation. D_p' is the normalized delay requirement of packet p_n, while L_p' is the normalized packet size. TR is the transfer rate of each node. We denote sim_i as the similarity of next-hop-selection between two data packet entities, which can be calculated by

$$sim_i = \frac{1}{\|P\|}\sum_{p_j \in P}\left(\|\boldsymbol{p_{n_r}} - \boldsymbol{p_{j_r}}\|_2 + \|\boldsymbol{p'_{n_c}} - \boldsymbol{p'_{j_c}}\|_2\right) \times \left[\left(\boldsymbol{v_{i_r}} - \boldsymbol{w_n}^{\mathrm{T}}\boldsymbol{v_{i_r}}\boldsymbol{w_n})\right) + \boldsymbol{d_n} - \left(\boldsymbol{p_{j_r}} - \boldsymbol{w_n}^{\mathrm{T}}\boldsymbol{p_{j_r}}\boldsymbol{w_n}\right)\right) + \left(\boldsymbol{l_{i_r}} - \boldsymbol{w_l}^{\mathrm{T}}\boldsymbol{l_{i_r}}\boldsymbol{l_n})\right) + \boldsymbol{d_l} - \left(\boldsymbol{p_{j_r}} - \boldsymbol{w_l}^{\mathrm{T}}\boldsymbol{p_{j_r}}\boldsymbol{w_l}\right)\right)\right], \quad (28)$$

where $\boldsymbol{w_n}$ and $\boldsymbol{d_n}$ denote the normal vector of hyperplane and relation vector of $NodePassedBy$ relation, while $\boldsymbol{w_l}$ and $\boldsymbol{d_l}$ denote those of $LinkPassedBy$ relation, respectively. If the packet p_n has similar vector representations with an existing packet p_j, and p_j has stronger connectivity with the next hop i, sim_i gets smaller and thus is more likely to be selected.

The route decision process of our algorithm is shown in Algorithm 2.

A. Time Computational Complexity Analysis

The time computational complexity of our KG-aided routing algorithm is $O((n+e+p)^2 + epoch \cdot N \cdot k + n \cdot k \cdot (n + e + p))$, where $n = \|V\|$ is the number of nodes; $e = \|E\|$ is the number of links; $p = \|P\|$ is the number of packets in the KG embedding process; $epoch$ is the time of iterations in Algorithm 1; N is the number of positive triples extracted from the network; k represents the dimension of vectors.

As mentioned before, the training process of our model consists of two stages, entity and relation extraction and KG embedding. In the stage of entities and relations extraction, the time complexity is decided by the number of nodes, links, and data packets, since the relations are predefined and can be regarded as constant. For each entity, we check all the other entities to extract their relations, and the time computational complexity is $O((n+e+p)^2)$. In KG embedding, we adopt MBGD to obtain the vector representations of all entities and relations. On each iteration, all positive triples are calculated once, so that the time computational complexity is $O(epoch \cdot N \cdot k)$.

After that, in the routing optimization process, we first calculate the vector representation of the incoming data packets, the time computational complexity of which is $O(k)$ if (19)–(20) have an exact solution, and $O(epoch \cdot k)$ if not. Then in route decision, we consecutively select the next nodes from the source node to the destination node, and the time computational complexity is $O(n \cdot k \cdot (n+e+p))$.

VI. Experiments

In this section, we simulate our proposed algorithm and analyze the results.

A. General Settings

We first introduce the general settings of our simulation and the parameters. The simulation topology of the experiment refers to the Iridium star system, which has 6 orbital planes, each orbital plane has 12 satellites, the orbital altitude is 780 kilometers, and each satellite has 4 inter-satellite links. We assume that all nodes can generate, forward, and process the arrival packets. To evaluate the dynamic characteristic of LEO networks, the probability of every two satellites being connected is 50% except for the last set of simulations. The delay of each node D_v follows a uniform distribution $D_v \sim U(5, 20)$ and the delay of each link follows $D_l \sim U(10, 40)$. The initial maximum queue length of each node follows a uniform distribution $QL_v^m \sim U(500, 1000)$ and maximum bandwidth follows $BW_l^m \sim U(500, 1000)$. For data packets, we randomly set delay requirements $D_p^{req} \sim U(100, 1000)$ and assume that the size of each packet follows $U(20, 100)$. The simulation is coded by Python 3.6, and runs on Windows 10 operating system with 8 GB memory.

B. Convergence Analysis

In KG construction process, we first randomly generate 1000 data packets in which the source node and destination node are randomly selected. The delay requirements and the size

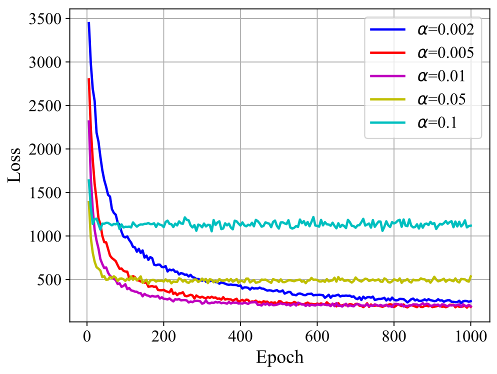

Fig. 5. Loss versus epoch in KG embedding.

of these packets both follow the same distribution in general settings. Besides, routes are previously calculated by the Floyd algorithm. These packets are considered as part of our training set and contribute to the extraction of triples. In the simulation, we extract 6694 positive triples from 1664 entities including nodes, links, and data packets. In KG embedding, we set margin $\gamma = 1$, $\eta = 1$, dimension $k = 50$, $b = 16$ and $C_1 = 1$.

Fig. 5 presents the trend of the loss functions of different learning rates α, i.e. (17) in the training process of KG embedding. As the number of epochs increases, the loss functions all first decrease sharply. This means that our model can effectively learn the characteristics of positive triples. Then, we notice that when the epoch is larger, the loss function tends to be constant after our model has well fit the triples. Since we adopt MBGD in training, the loss function may increase at specific epochs, but generally decreases as the epoch becomes larger. We can also observe that the rate of convergence is faster with a larger α, while an overlarge α may cause underfitting, as the loss functions of $\alpha = 0.05$ and $\alpha = 0.1$, which converge at a larger constant. To avoid underfitting and overfitting, we set $epochs = 500$ and $\alpha = 0.01$ in our training process on this specific network.

The embedding result of vectors $\boldsymbol{v_r}$, $\boldsymbol{l_r}$, $\boldsymbol{p_r}$, $\boldsymbol{w_r}$, $\boldsymbol{d_r}$ are all k-dimensional normalized vectors where $k = 50$. Here, we show several of them as follows

$$\begin{aligned}\boldsymbol{v_{0r}} &= [-0.1776, \quad -0.3709, -0.1586, \ldots, \quad 0.1363]\\ \boldsymbol{l_{0r}} &= [0.0742, \quad -0.1973, -0.0819, \ldots, \quad 0.0098]\\ \boldsymbol{p_{0r}} &= [-0.0688, \quad 0.0572, -0.1280, \ldots, \quad 0.1532]\end{aligned}$$

C. Performance Analysis

In routing calculation, we set $C_2 = 1$ and $\beta = 0.5$, where C_2 denotes the weight parameter for the normalization and β denotes the weight factor to adjust the weight of the next node. We first perform two experiments assuming that the arrival rate of the data packets follows a Poisson distribution with $\lambda = 0.5$ and $\lambda = 1.5$, respectively. We compared our proposed algorithm with benchmarks Dijkstra algorithm [31] and Floyd algorithm [32]. The results are shown in Figs. 6 and 7.

In the first observation, we assume that $\lambda = 0.5$, indicating that the network is filled with a small number of data packets. As shown in Fig. 6(a), we notice that when the network is not busy, the proposed algorithm can successfully calculate feasible routes for all packets, while the Floyd algorithm and Dijkstra algorithm fails to steer about 1.2% of the data packets. This is because the two baselines calculate the shortest paths from the source node to the destination node, in neglect of other possible paths, which leads to network congestions in bottleneck nodes and links. KG-SDNRL algorithm selects routes in consideration of available resources based on their vectors and thus avoids congestions when the network is not busy. Fig. 6(b) illustrates the average throughput of three algorithms. As the experiment goes on, the average throughput of the Dijkstra algorithm decreases while that of the KG algorithm increases. The KG-assisted routing strategy allows throughput to increase with simulation time, making the network more intelligent and efficient. Since the arrival of user packets follows a Poisson distribution, a small difference in average throughput is reasonable. It can be proved the expectation of average throughput $\mathbb{E}(throughput) = \lambda$ if the packet loss ratio is 0. Since the Dijkstra algorithm has a packet loss ratio of about 1.2%, its average throughput is smaller. Fig. 6(c) shows that the average delays under the three algorithms all increase with the simulation, while the delay increases most gently with the assistance of KG-SDNRL. This is because the proposed KG architecture embeds network entities and relations into vectors before we calculate routing decisions, including the data packets that have already been transmitted. On the one hand, vector representation can capture complex information about the network, such as paths and high-order proximity, which contributes to the route calculations. On the other hand, the data packets offer route information that contains feasible routing paths.

Furthermore, we evaluate the performance of the proposed algorithm under an overloaded network with $\lambda = 1.5$. As depicted in Fig. 7(a), the average packet loss rate of the Dijkstra algorithm is around 10%, and that of the Floyd algorithm is about 8%. Compared to the result in Fig. 7(a), we can observe that our proposed algorithm can handle more data packets with a lower packet loss rate, on account that the KG architecture embeds network entities into low dimensional vectors, which contain relational information such as paths that contributes to routing decisions to avoid congestions. Thus, our KG routing algorithm is capable of handling complex and overloaded networks with more data transmitted. We further observe the average throughput of three algorithms as shown in Fig. 7(b). The KG routing algorithm is the only one where the throughput increases steadily as the simulation progresses. Fig. 7(c) illustrates that three algorithms have a similar average delay under the busier network.

We further compare the performance of widely used routing strategies in LEO networks with our proposed algorithm as shown in Fig. 8. We use the DT-DVTR algorithm as our baseline, where the network is divided into time slices of equal period length, thereby turning the dynamic satellite network into static slices, and then using the Dijkstra algorithm. Considering a dynamic LEO network with a normal workload, we can observe that the KG-SDNRL algorithm has the lowest packet loss rate and largest average throughput. The above three groups of simulation results show that the KG-SDNRL routing algorithm has a lower packet loss rate and higher throughput in LEO dynamic satellite network scenario.

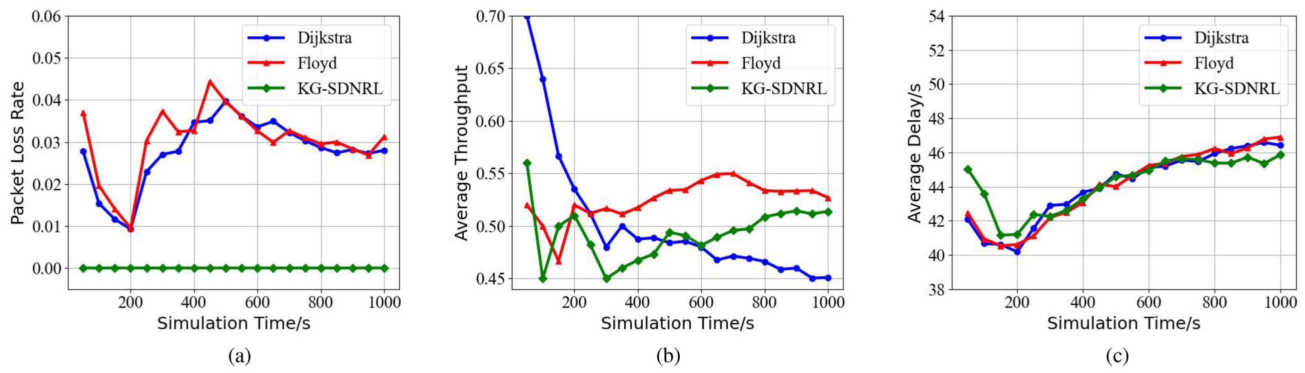

Fig. 6. Packet loss ratio, average throughput, and average delay with λ = 0.5. (a) Packet Loss Ratio. (b) Average Throughput. (c) Average Delay.

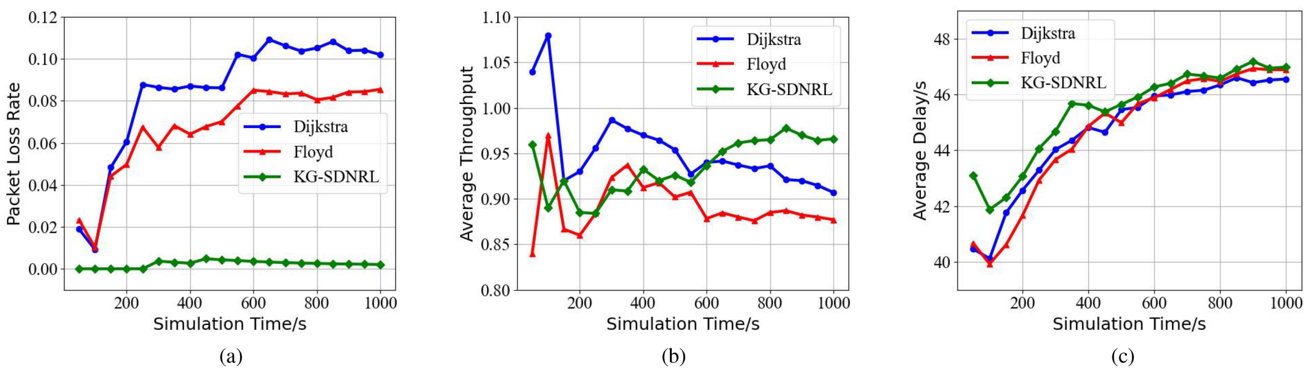

Fig. 7. Packet loss ratio, average throughput, and average delay with λ = 1.5. (a) Packet Loss Ratio. (b) Average Throughput. (c) Average Delay.

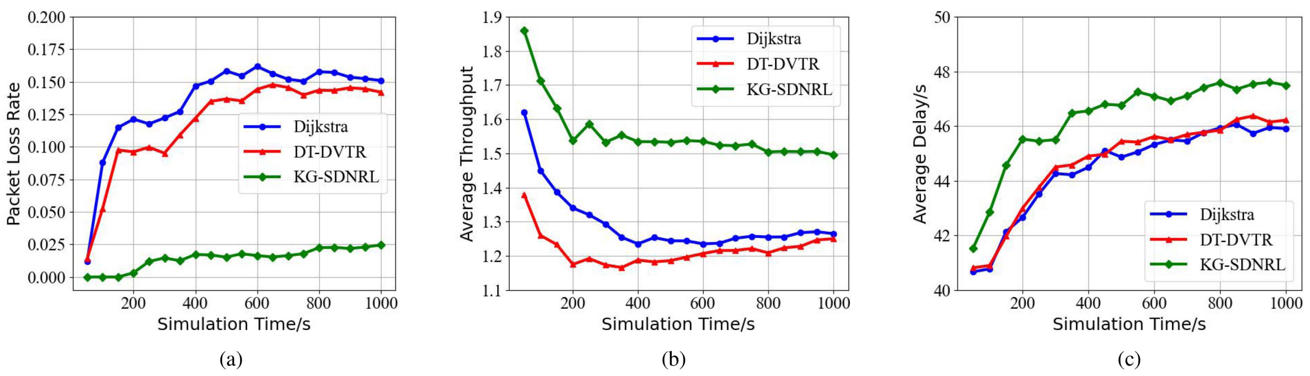

Fig. 8. Performance of three LEO routing strategies. (a) Packet Loss Ratio. (b) Average Throughput. (c) Average Delay.

To better illustrate the scalability of our algorithm, we further evaluate the network consisting of different numbers of substrate nodes and compared their average delay. As shown in Fig. 9, the average delay varies at the beginning of the simulation since only a small number of user packets have arrived. Our work focuses on the dynamics of the satellite network topology, so we observe the performance of the algorithm under different topological node counts. As the simulation goes on, it gradually converges to a certain value of around 37. Moreover, the average delay of user packets becomes longer as the number of nodes increases, for the simple reason that the paths tend to be longer with more nodes. However, it can be observed that the difference between these results is small. Since network entities and relations are embedded into low-dimensional vectors, the expansion of the network affects little on the network representation, and also affects little on the average delay. This proves that our algorithm is capable of deciding feasible routes effectively in larger-scale networks. Finally, we carry out experiments on networks with worse connectivity, i.e. substrate nodes are less likely to connect to the others. We simulate this by setting the probability of every two nodes to be connected as 25%, which is half of that in the previous simulations. This results in a smaller number of triples extracted and thus may be more difficult for our model to capture network characteristics. Fig. 10 shows the average

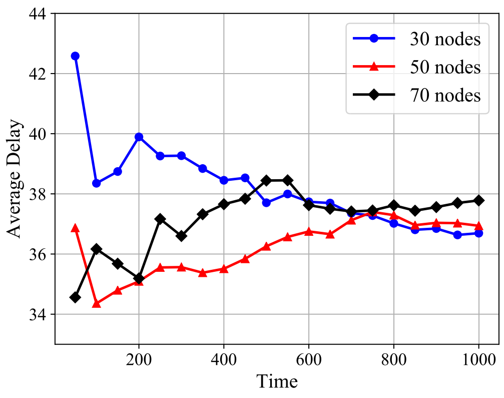

Fig. 9. Average delay on a different number of substrate nodes.

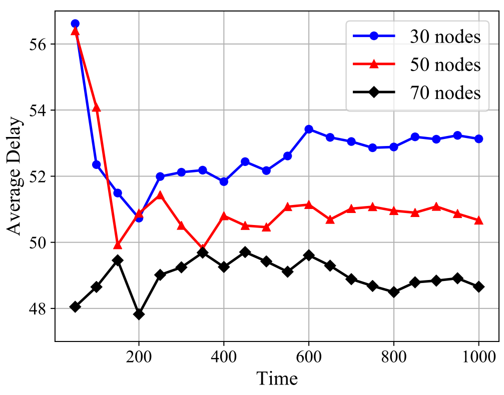

Fig. 10. Average delay on a different number of substrate nodes when connectivity becomes worse.

delay on three networks with different numbers of substrate nodes. It can be seen that as the number of nodes increases, more information is used to construct the knowledge graph, and the routing performance of the network is better. For LEO satellite networks, the results demonstrate the high scalability of our proposed algorithm. The KG-SDNRL algorithm applies to the increasing number of satellites and the dynamic change of topology.

As the above analysis, the average delay decreases with the increasing number of substrate nodes. To better analyze the cause of this result, we calculate the average number of hops of the routing results, which is depicted in Fig. 11. Fig. 11 illustrates that the average number of hops of the network with 30 nodes is the largest at around 1.8 hops per packet, while that of the networks with 50 and 70 nodes are both around 1.77 hops per packet. We check the routing results of the packets and find out that several packets are transferred through 6 nodes in the 30-node network. If the nodes are connected sparsely in smaller networks, it is more likely that some of the nodes are away from each other since fewer transferring nodes can be selected in routing calculations. In larger networks with more substrate nodes, the average number of hops becomes slightly smaller as there are more links, and fewer packets are transferred through extremely long routes. Furthermore, it can be observed in Fig. 10 that the average delay of the network with 50 nodes is larger than that with 70 nodes, and their average number of hops is

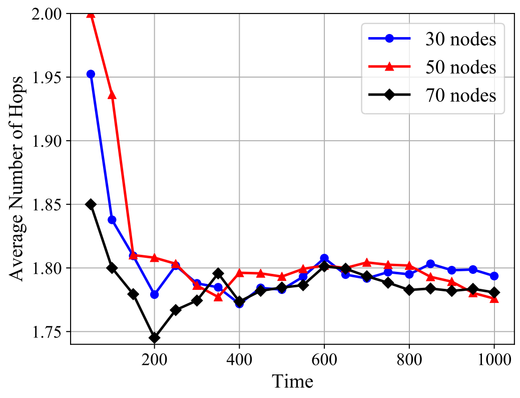

Fig. 11. Average number of hops on different numbers of substrate nodes when connectivity becomes worse.

both around 1.77 hops per packet. The reason why the delay performance may decrease slightly in sparse matrices is that knowledge graph methods are based on topological information. At the beginning of algorithm learning, the knowledge of sparse network topology is relatively small, which will affect the performance of output results. This finally affects the performance of our algorithm on a small-scale network or a sparsely connected one. Moreover, the performance of throughput and packet loss rate is satisfactory even in sparse network topology. As a result, few triples are extracted in the entity extraction process, which is not beneficial to the training of our model. Based on the above analysis, KG-SDNRL is an effective routing algorithm that can calculate an available path from the source node to the destination node, both meeting user's QoS requirements and maintaining network functionalities.

VII. Conclusion

In this paper, we adopt a modified KG architecture to represent an LEO satellite network by embedding nodes, links, data packets, and their relations which represent connectivity into low-dimensional vectors. In the training process, we first extract triples from the entire network to capture the relational features such as topology in the network. Then the vector representations of entities and relations are trained through MBGD. In this way, we enhance the representation ability of the network structure. Besides, to facilitate the routing process, we first calculate the vector representation of the incoming packets. After that, we calculate a feasible route according to these vectors for each user packet by successively selecting nodes and links. The experiment result shows that our algorithm has a lower packet loss rate and average delay, which indicates that KG is capable of representing network structure and has great potential in network optimizations.

In the future, we will mainly focus our attention on the following two aspects. 1) KG is one of the promising techniques to model relational data structure, while there are also many other approaches in NRL. We will try to vectorize the network by adopting another KG embedding scheme or utilizing another vectorization model and try to figure out a better way to model a network. 2) After embedding network entities to low-dimensional vectors, we can also adopt an ML-based routing model to capture these extracted attributes and obtain routing schemes for complex objectives.

References

[1] Z. Qu, G. Zhang, H. Cao, and J. Xie, "LEO satellite constellation for Internet of Things," *IEEE Access*, vol. 5, pp. 18391–18401, 2017.

[2] Y. Xiao et al., "A LEO satellite network capacity model for topology and routing algorithm analysis," in *Proc. IEEE 14th Int. Wireless Commun. Mobile Comput. Conf.*, 2018, pp. 1431–1436.

[3] Y. Su, Y. Liu, Y. Zhou, J. Yuan, H. Cao, and J. Shi, "Broadband LEO satellite communications: Architectures and key technologies," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 55–61, Apr. 2019.

[4] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in SDN," in *Proc. ACM Symp. SDN Res.*, San Jose, CA, USA, 2019, pp. 140–151.

[5] J. Rexford, *Route Optimization in IP Networks*. Boston, MA, USA: Springer, 2006, pp. 679–700.

[6] Q. I. Xiaogang, M. A. Jiulong, W. U. Dan, L. I. U. Lifang, and H. U. Shaolin, "A survey of routing techniques for satellite networks," *J. Commun. Inf. Netw.*, vol. 1, no. 4, pp. 66–85, 2016.

[7] J. Jin, F. Tian, Z. Yang, H. Di, and G. Li, "A disruption tolerant distributed routing algorithm in LEO satellite networks," *Appl. Sci.*, vol. 12, no. 8, 2022, Art. no. 3802.

[8] C. Garibotto, A. Sciarrone, F. Lavagetto, L. Pronzati, A. Baljak, and G. Tagliabue, "Performance analysis of a IoT-based personal vocal assistant for cruise ships over satellite networks," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14857–14866, Aug. 2022.

[9] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2020.

[10] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Melbourne, Australia, 2015, pp. 891–900.

[11] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, Dec. 2017.

[12] L. Heck and H. Huang, "Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, Atlanta, Georgia, USA, 2014, pp. 597–601.

[13] C. Wang, X. Ma, J. Chen, and J. Chen, "Information extraction and knowledge graph construction from geoscience literature," *Comput. Geosci.*, vol. 112, pp. 112–120, Dec. 2017.

[14] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Melbourne, Australia, 2019, pp. 105–113.

[15] L. Cui, F. R. Yu, and Q. Yan, "When Big Data meets software-defined networking: SDN for Big Data and Big Data for SDN," *IEEE Netw.*, vol. 30, no. 1, pp. 58–65, Jan./ Feb. 2016.

[16] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Commun. Surv. Tut.*, vol. 20, no. 1, pp. 388–415, Jan.–Mar. 2018.

[17] Y. Gong, H. Yao, J. Wang, L. Jiang, and F. R. Yu, "Multi-agent driven resource allocation and interference management for deep edge networks," *IEEE Trans. Veh. Technol.*, vol. 71 no. 2, pp. 2018–2030, Feb. 2022.

[18] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Mini-batch gradient descent: Faster convergence under data sparsity," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, Melbourne, Australia, 2017, pp. 2880–2887.

[19] M. Werner, "A dynamic routing concept for ATM-based satellite personal communication networks," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 8, pp. 1636–1648, Oct. 1997.

[20] Y. Hashimoto and B. Sarikaya, "Design of IP-based routing in a LEO satellite network," in *Proc. 3rd Int. Workshop Satell.-Based Inf. Serv.*, 1998, pp. 81–88.

[21] E. Ekici, I. F. Akyildiz, and M. D. Bender, "Datagram routing algorithm for LEO satellite networks," in *Proc. IEEE Conf. Comput. Commun., 19th Annu. Joint Conf. IEEE Comput. Commun. Societies*, 2000, vol. 2, pp. 500–508.

[22] Y. Zhang et al., "A novel QoS routing method for LEO rosette constellation network," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Commun.; Dependable, Autonomic Secure Comput.; Pervasive Intell. Comput.*, 2015, pp. 525–531.

[23] P. Wang, B. Di, and L. Song, "Multi-layer LEO satellite constellation design for seamless global coverage," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.

[24] Z. Wu, G. Hu, F. Jin, Y. Song, Y. Fu, and G. Ni, "A novel routing design in the ip-based GEO/LEO hybrid satellite networks," *Int. J. Satell. Commun. Netw.*, vol. 35, no. 3, pp. 179–199, 2017.

[25] H. Tan and L. Zhu, "A novel routing algorithm based on virtual topology snapshot in LEO satellite networks," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, 2014, pp. 357–361.

[26] Y. Liu and C. Liu, "Distributed dynamic routing algorithm for satellite constellation," in *Proc. IEEE 10th Int. Conf. Commun. Softw. Netw.*, 2018, pp. 300–304.

[27] H. Yao, T. Mai, C. Jiang, L. Kuang, and S. Guo, "AI routers & network mind: A hybrid machine learning paradigm for packet routing," *IEEE Comput. Intell. Mag.*, vol. 14, no. 4, pp. 21–30, Nov. 2019.

[28] Y. Zhu, L. Qian,L. Ding, F. Yang, C. Zhi, and T. Song, "Software defined routing algorithm in LEO satellite networks," in *Proc. IEEE Int. Conf. Elect. Eng. Inform.*, 2017, pp. 257–262.

[29] C. Han, H. Yao, T. Mai, N. Zhang, and M. Guizani, "QMIX aided routing in social-based delay-tolerant networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 1952–1963, Feb. 2022.

[30] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 16, no. 8, pp. 5444–5455, Aug. 2020.

[31] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[32] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, Jun. 1962, Art. no. 345.

[33] P. A. Humblet, "Another adaptive distributed shortest path algorithm," *IEEE Trans. Commun.*, vol. 39, no. 6, pp. 995–1003, Jun. 1991.

[34] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Traffic engineering enhancement by progressive migration to SDN," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 438–441, Mar. 2018.

[35] Y. S. Yu and C. H. Ke, "Genetic algorithm-based routing method for enhanced video delivery over software defined networks," *Int. J. Commun. Syst.*, vol. 31, Aug. 2017, Art. no. e3391.

[36] M. K. Awad, M. El-Shafei, T. Dimitriou, Y. Rafique, M. Baidas, and A. Alhusaini, "Power-efficient routing for SDN with discrete link rates and size-limited flow tables: A tree-based particle swarm optimization approach," *Int. J. Netw. Manage.*, vol. 27, no. 5, Mar. 2017, Art. no. e1972.

[37] H. Yao, X. Yuan, P. Zhang, J. Wang, C. Jiang, and M. Guizani, "Machine learning aided load balance routing scheme considering queue utilization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7987–7999, Aug. 2019.

[38] A. Jindal, G. S. Aujla, N. Kumar, R. Chaudhary, M. S. Obaidat, and I. You, "SeDaTiVe: SDN-enabled deep learning architecture for network traffic control in vehicular cyber-physical systems," *IEEE Netw.*, vol. 32, no. 6, pp. 66–73, Nov./ Dec. 2018.

[39] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: Optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 64533–64539, 2018.

[40] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep Q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5871–5883, Jun. 2019.

[41] H. Yao, H. Liu, P. Zhang, S. Wu, C. Jiang, and S. Guo, "A learning-based approach to intra-domain QoS routing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6718–6730, Jun. 2020.

[42] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 2787–2795, 2013.

[43] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th AAAI Conf. Artif. Intell.*, Québec City, Québec, Canada, 2014, pp. 1112–1119.

[44] H. Yao, T. Mai, X. Xu, P. Zhang, M. Li, and Y. Liu, "NetworkAI: An intelligent network architecture for self-learning control strategies in software defined networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4319–4327, Dec. 2018.

[45] J. Wang, C. Jiang, H. Zhang, Y. Ren, K. C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to Pareto-optimal wireless networks," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 1472–1514, Jul.–Sep. 2020.

[46] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 1067–1077.

Chenxi Li received the bachelor's degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2018, where he is currently working toward the master's degree. His research interests include network virtualization and future network architecture.

Wenji He (Student Member, IEEE) received the bachelor's degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2021, where she is currently working toward the Ph.D. degree. Her research interests include computer working and the security of the Internet.

Haipeng Yao (Senior Member, IEEE) received the Ph.D. degree from the Department of Telecommunication Engineering from the University of Beijing University of Posts and Telecommunications, Beijing, China, in 2011. He is currently the Professor with the Beijing University of Posts and Telecommunications. His research interests include future network architecture, network artificial intelligence, networking, space-terrestrial integrated network, network resource allocation and dedicated networks. He has authored or coauthored more than 100 papers in prestigious peer-reviewed journals and conferences. He was the Editor of IEEE NETWORK, IEEE ACCESS, and the Guest Editor of the IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY and SPRINGER JOURNAL OF NETWORK AND SYSTEMS MANAGEMENT. He was a Member of the Technical Program Committee and the Symposium Chair for a number of international conferences, including IWCMC 2019 Symposium Chair, ACM TUR-C SIGSAC2020 Publication Chair.

Tianle Mai (Student Member, IEEE) is currently working toward the Ph.D. degree with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. His main research interests include artificial intelligence, multi-agent system, and future network architecture.

Jingjing Wang (Senior Member, IEEE) received the B.S. degree (with highest Hons.) in electronic information engineering from the Dalian University of Technology, Dalian, China, in 2014, and the Ph.D. degree (with highest Hons.) in information and communication engineering from Tsinghua University, Beijing, China, in 2019, From 2017 to 2018, he visited the Next Generation Wireless Group chaired by Prof. Lajos Hanzo, University of Southampton, Southampton, U.K. He is currently an Associate Professor with the School of Cyber Science and Technology, Beihang University, Beijing, China. His research interests include AI enhanced next-generation wireless networks, UAV swarm intelligence, and confrontation. He has authored or coauthored more than 100 IEEE Journal/Conference papers. He was the recipient of the Best Journal Paper Award of IEEE ComSoc Technical Committee on Green Communications and Computing in 2018, the Best Paper Award of IEEE ICC and IWCMC in 2019.

Song Guo (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Ottawa, Ottawa, ON, Canada. He is currently a Full Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. Prior to joining PolyU, he was a Full Professor with the University of Aizu, Aizuwakamatsu, Japan. His research interests include the areas of cloud and green computing, Big Data, wireless networks, and cyber-physical systems. He has authored or coauthored more than 300 conference and journal papers in these areas and was the recipient of multiple Best Paper awards from IEEE/ACM conferences. His research has been sponsored by JSPS, JST, MIC, NSF, NSFC, and industrial companies. He was the Editor of several journals, including IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, *IEEE Communications Magazine*, and *Wireless Networks*. He has been actively participating in international conferences as a General Chair and TPC Chair. He is a Senior Member of ACM, and an IEEE Communications Society Distinguished Lecturer.